THE CONNECTION MACHINE™ COMPUTER
A Natural Fit To Applications Needs

THE CONNECTION MACHINE™ COMPUTER
A Natural Fit To Applications Needs

The Connection Machine computer is a massively parallel system which reconfigures itself to match the natural structure of applications exactly, while providing computing power in excess of 1000 MIPS. Unique among parallel architectures, the system allows the connections among its processors to be altered at will. It is the next major advance in large scale computing.

The Connection Machine system is unique in its ability to adapt to the natural structure of the problem it is addressing. There are three steps in setting up the system to match the target problem. The first is to assign a processor to each individual problem element. The second is to arrange the interconnections among the processors to match the way individual problem elements interact. The third is to execute the calculations on all these interconnected elements concurrently.

A Processor For Each Problem Element

All large scale computer systems have to deal with overwhelming numbers of problem elements. In science and engineering, it is the quest for detail that creates huge numbers of elements. A simulation of an airplane wing, for example, may involve the tracking of turbulence at 200,000 individual points around the wing surface. A weather simulation may operate on 250,000 points in the atmosphere. In artificial intelligence and knowledge representation applications, it is the number of alternatives that raise the complexity level. A system may be required to choose intelligently from among 10,000 hypotheses.

Traditional large scale systems have dealt with detail by sequentially cycling a small number of processors (1 to 4) through all the problem elements as quickly as possible. That is why they are called serial systems. The Connection Machine dynamic computer employs a more powerful and simpler way. It assigns a complete processor to each problem element, whether there are 50,000 of them or a million.

Matching the Way Problem Elements Interact

Individual elements of a problem do not exist in isolation. They interact and influence the state of other elements. It is, in fact, the way they interact that is at the core of the problem. In a VLSI simulation, for example, the fact that a transistor has changed state is only one part of the situation. The effect of this state change on the other transistors it is connected to is the really important element.

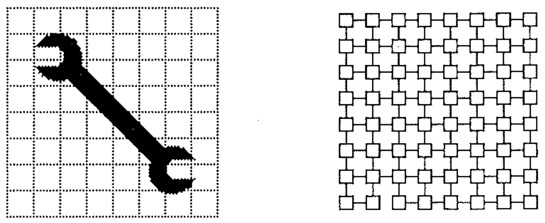

Figure 1:
The Connection Machine system configured for image processing.

No two problems have the same internal structure. Each has its own characteristic interaction paths, or "topology." The Connection Machine system, unique among large scale systems, adapts itself accordingly. Some problems, like image processing, have very regular connections. Elements of the image are held in "pixels." Processing the image involves large numbers of interactions between neighboring pixels. Therefore, for this application, the system organizes itself as a grid. Each processor is connected to the processor above, below, and to the sides.

VLSI simulation is a very different type of problem. The connection arrangement represents the wiring of the circuit; it is unique to an individual chip. A transistor may be connected to the one next door, the one clear across the chip, or both. Again, the Connection Machine system organizes itself accordingly. A processor is assigned to each transistor. A connection is established for each wire.

Fast Fourier Transform (FFT) applications also have a distinctive topology, which is matched exactly by the system. The Connection Machine system is very efficient at FFT's. Artificial intelligence, natural language processing, and knowledge representation are still a fourth important class of problems with a distinctive topology. Here problem elements link together in a variety of tree or graph structures. Each branch point can, for example, represent an alternative decision.

Connection Machine processors are not pre-wired in any of these topologies. Each of them is accommodated with equal ease, because all linkages are software controlled. Two processors that know each other's addresses are thereby linked together. System-wide message flow is handled by a 3 gigabit per second message routing system.

Computing in Parallel

All giant, high performance computers, parallel or serial, contain about a square meter of silicon circuitry. The difference is in how that square meter is utilized. A serial computer has a single central processor which uses about 5% of the

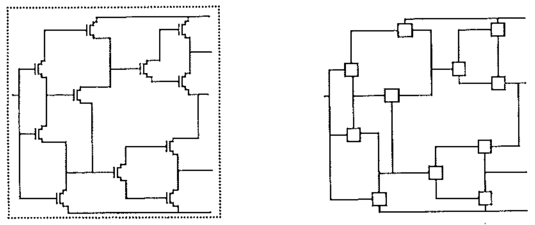

Figure 2:
The Connection Machine system configured for VLSI simulation.

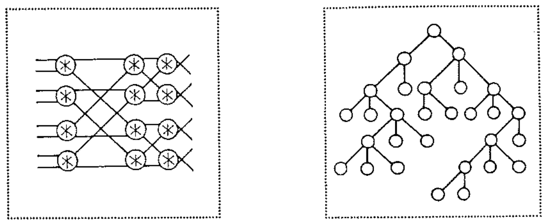

Figure 3:
FFT's and Trees : two more topologies that are matched exactly by the Connection Machine system.

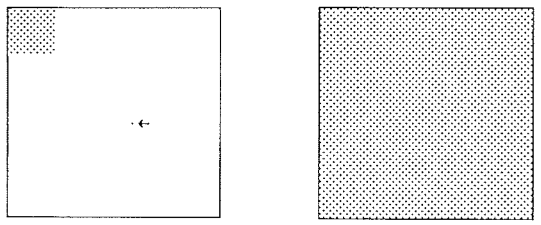

Figure 4:
(a) A serial architecture has a busy cpu, but the rest of the silicon, the memory, is hardly busy at all.
(b) The Connection Machine system has massive numbers of processors and hence keeps more of its circuitry busy.

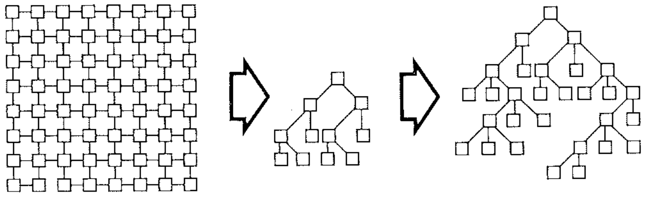

Figure 5:
A machine vision application requires a grid structure for initial pixel analysis, then a series of tree structures as the system works to identify the objects detected. The changes are made dynamically, simply by changing addresses in individual processors.

silicon area. The rest is used for memory.

This serial architecture, the von Neumann architecture, originated well before the availability of solid-state memory. When central processing modules and memory arrays were made of completely different components, there was no easy way to trade them off or make them physically close. The most practical approach was to house the processing logic and the memory separately, connected by a memory bus. The consequence is that the memory as a whole is not kept busy at all; only one location is active at a time. This limitation is known as the von Neumann bottleneck.

Recognizing that processors and memory are now made from the same silicon, the Connection Machine system's designers included huge numbers of processors. Furthermore, they distributed these processors throughout the memory. In this way tens of thousands of memory locations are busy at a time, not just one. By keeping more of its square meter of circuitry busy all the time, the Connection Machine system outperforms serial systems by a wide margin and is more cost-effective as well.

The one-to-one match of processors and problem elements in the Connection Machine system is implemented at two distinct levels. First, there are the 65,536 physical processors in the machine. Each is a complete computer with its own memory. All 65,536 computers fetch data and compute in parallel. At the second level of implementation, individual physical processors are partitioned by software into virtual processors. The enormous total horsepower of the system makes virtual processing feasible for problems with as many as a million elements.

Dynamic Reconfiguration

The Connection Machine reconfiguration capability is especially valuable for problems whose structure changes dynamically. New processor connections may be established at any time, simply by storing the new addresses. A single Connection Machine system can often replace multiple special-purpose machines, each of which can only handle one stage of the problem.

Using the Connection Machine System

The Connection Machine system is not complicated to use. Both the hardware and the software employ a small set of very powerful tools. The speed of the system comes from the way these tools are used.

Access to all Connection Machine capabilities is via a front-end processor, either a Symbolics 3600[1] or a Digital VAX[2]. This front end provides the operating system environment, including terminal interaction and file management. It also holds the applications programs. Applications may be written in either C* ™ (a Connection Machine extension of C) or CM Lisp™ (a Connection Machine extension of Lisp.)

The first step in solving a large-scale problem is to break it down into its individual problem elements and assign one element to a processor. If there are 1,000,000 such problem elements, the machine is configured for a million virtual processors. The command to do this is contained in the applications program which resides in the host machine. It is transferred to the Connection Machine hardware for execution.

The next step is to initialize each virtual processor to the state it will be in when execution begins. Each problem element will have a set of numbers that characterizes its initial state. It will also have a list of the other problem elements that it interacts with. This list will be in the form of pointers to the processors which contain these problem elements. If the problem is a VLSI circuit simulation, each element is a transistor. The state information indicates whether that transistor is on or off. The pointers tell which other transistors are wired to the output of this one.

Each virtual processor in the system has its own memory. When the Connection Machine hardware is operating as a physical machine with 65,536 processors, each processor has 4096 bits of memory, for a system total of 32 Mbytes. If the system is set up for a greater number of virtual proces-

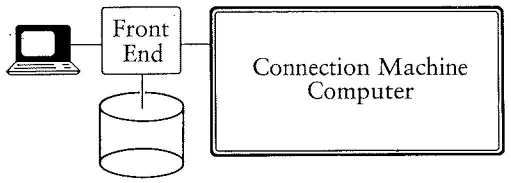

Figure 6:
The Connection Machine Front End

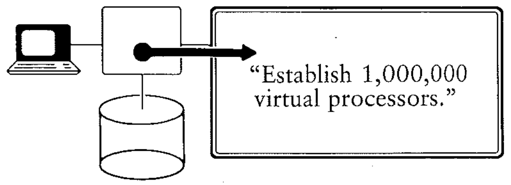

Figure 7:
Setting up the Virtual Processors.

[1]"Symbolics 3600" is a trademark of Symbolics, Inc.
[2]"VAX" is a trademark of Digital Equipment Corp.

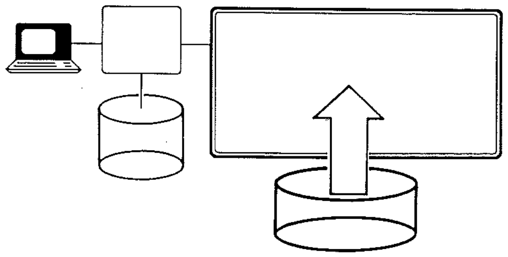

Figure 8:
Loading Data into the Connection Machine Memory.

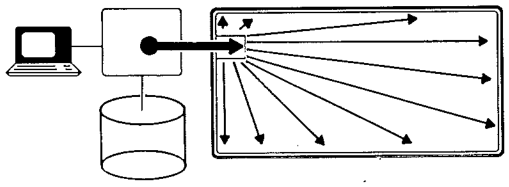

Figure 9:
Parallel Execution of Instructions.

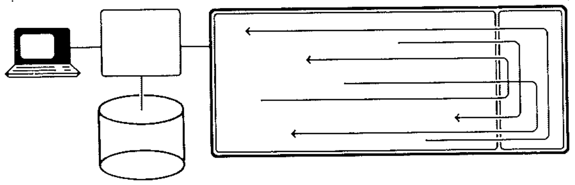

Figure 10:
A Message Communication Cycle Using the Router.

sors, the memory per processor is correspondingly smaller.

Loading of the initial state information proceeds in one of two ways. If the amount of data is small, it is loaded directly from the front end. All 32 Mbytes of Connection Machine memory are dual-ported and hence directly accessible by both machines. If the amount of data is large, such as the initialization data for a million-element simulation, it is loaded directly from the system's swapping disk. This 1.2-gigabyte disk system transfers to and from the Connection Machine memory at the rate of 500 megabits per second.

Once the problem elements are set up, program execution proceeds under control of the front end. The program, stored on the front end's disk, is loaded and executed. If the instructions are serial (i.e., they only need to be executed once) they are handled directly by the front end processor. Parallel instructions are passed to the Connection Machine hardware for execution. A microcontroller receives the instructions, expands them into Connection Machine "nano-instructions" and broadcasts these to all processors in the system.

Parallel instructions are of two types. The first type computes: manipulating information within each virtual processor. In a simulation, the computation would update the state of each problem element. The second type of instruction communicates: passing information from one problem element to another. Communication is a parallel instruction; all processors may send messages in parallel. An extremely high speed pipelined router manages message delivery.

From an applications programming perspective, this is how the Connection Machine system operates. It is simple to use because it fits so naturally with the problem to be solved. But backing up that simplicity is 1000 MIPS of computing power.

Summarizing the Connection Machine System

Every element of the Connection Machine system contributes to its applications orientation. Everything focuses on making large-scale problem solving easier. Dynamic reconfiguration sets the computer up the same way the problem is set up. Virtual processors allow a 1-on-1 matching of problem elements and processors. 65,000 physical processors provide the raw computing power (in excess of 1000 MIPS) to solve giant problems quickly. The speed of the router (over 3 billion bits per second) accommodates problems with massive amounts of intercommunication. And yet all this power is delivered with just a handful of new constructs in CM Lisp and C*.

Thinking Machines Corporation believes that advances in large scale computing require both power and simplicity. The power gives access to the needed detail. The simplicity allows the user to forget about the components of the computer and focus on the components of the problem. The magic of the Connection Machine system is that it provides both.

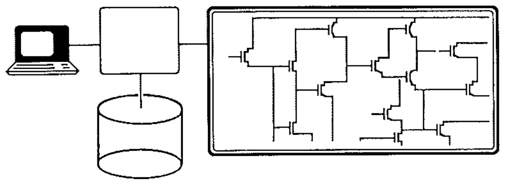
Figure 11:
The Connection Machine system is easily set up for VLSI simulation.

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts
02142-1214

(617) 876-1111